SQL Problems - 1

1. JOINS Problems

a. Analyzing Customer Orders

- *Problem*: Retrieve all customers who placed an order along with their order details. If a customer hasn't placed an order, show their details with NULL values for order details.
  - **Concept**: LEFT JOIN
- *Bonus*: Identify customers who haven't placed any orders.
  - **Concept**: LEFT JOIN with `WHERE order_id IS NULL`

b. Linking Payments to Orders

- *Problem*: Find all orders and their associated payments. Show orders that haven't been paid yet.
  - **Concept**: LEFT JOIN between orders and payments

c. Cross-Selling Analysis

- *Problem*: Generate a list of customers, their orders, and the payment method used. For customers without payments, show "No Payment".
  - **Concept**: LEFT JOIN + CASE WHEN

d. Aggregating Total Sales

- *Problem*: Calculate total sales (sum of `order_amount`) for each customer by joining `orders` and `customers`.
  - **Concept**: INNER JOIN with `GROUP BY`

e. Multi-Table Joins

- *Problem*: Find the total payment amount for each customer who has made at least one payment. Combine data from `customers`, `orders`, and `payments`.
  - **Concept**: Joins across 3 tables

---

2. NULL Functions Problems

a. Handling Missing Data

- *Problem*: Identify all orders that haven't been paid and replace the NULL payment amount with `0`.
  - **Concept**: COALESCE()

b. Null Customer Cities

- *Problem*: Identify customers whose city is missing and replace it with "Unknown".
  - **Concept**: COALESCE()

c. Default Payment Method

- *Problem*: For all orders without a recorded payment method, display "Pending Payment".
  - **Concept**: COALESCE() on `payment_method`

---

3. CASE WHEN Problems

a. Categorizing Orders

- *Problem*: Categorize orders into "Small" (order amount < 100), "Medium" (100-300), and "Large" (>300).
  - **Concept**: CASE WHEN

b. High-Value Customers

- *Problem*: Mark customers as "High Value" if their total orders exceed $10,000; otherwise, mark them as "Regular".
  - **Concept**: CASE WHEN with aggregation (`GROUP BY` + HAVING)

c. Payment Completion Status

- *Problem*: For each order, show "Paid" if a payment exists and "Unpaid" otherwise.
  - **Concept**: CASE WHEN + LEFT JOIN

d. Revenue Breakdown by State

- *Problem*: Calculate total revenue by state and classify states as "Low Revenue" (<$50,000), "Moderate Revenue" ($50,000-$100,000), or "High Revenue" (>$100,000).
  - **Concept**: CASE WHEN + Aggregation

e. Customer Lifetime Value

- *Problem*: Categorize customers based on their total order value:
  - "Bronze" (less than $5,000)
  - "Silver" ($5,000-$10,000)
  - "Gold" (more than $10,000)
  - **Concept**: CASE WHEN with aggregation

Complex Problems Combining Concepts

1. Unpaid Orders Report

- *Problem*: Generate a report of all unpaid orders with customer names, cities, and the missing payment amount set to `0`.
  - **Concepts**: LEFT JOIN + COALESCE() + CASE WHEN

2. Customer and Revenue Insights

- *Problem*: For each customer, calculate the total revenue from their orders and classify them as "Active" or "Inactive" based on whether they placed an order.
  - **Concepts**: Aggregation + LEFT JOIN + CASE WHEN

3. Payment Recovery Report

- *Problem*: Identify all orders with partial payments. Show the amount still due for each order and classify the status as "Partial" or "Fully Paid".
    - **Concepts**: JOINS + COALESCE() + CASE WHEN

4. Monthly Sales Trends

- *Problem*: Calculate monthly sales and classify each month as "Low Sales" (<$50,000) or "High Sales" (>$50,000).
    - **Concepts**: Aggregation + CASE WHEN